HANOI UNIVERSITY OF SCIENCE AND TECHONOLOGY

SCHOOL OF INFORMATION COMMUNICATION TECHNOLOGY

Scientific Computing Capstone project report:

Using MATLAB to solve the 2D Heat Equation

Supervised by:

PhD. Vu Van Thieu

Group members:

Tran Duc Le Huy - 20225978

Nguyen Vu Dung - 20225964

Pham Phuong Huy - 20225977

Phan Dao Minh Quan - 20225993

Vo Thanh Vinh - 20226074

Hanoi - Vietnam

May 2024

Contents

Abstract

This project focuses on using MATLAB to solve the two-dimensional (2D) heat equation, a fundamental partial differential equation (PDE) describing the distribution of heat (or temperature) in a given region over time. The solution involves numerical methods, specifically the finite difference method, to approximate the temperature distribution in a thin rectangular plate.

1 Introduction

The heat equation is a vital tool in mathematical physics and engineering for modeling the distribution of temperature in a given domain. In two dimensions, it can be applied to understand heat distribution in a thin plate, which has practical applications in various fields such as material science, mechanical engineering, and environmental science. This project aims to use MATLAB to solve the 2D heat equation, providing a detailed methodology and analysis of the results.

2 Problem Statement

The objective is to solve the 2D heat equation for a thin rectangular plate. The equation is given by:

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial z^2} \right) \tag{1}$$

where:

- T is the temperature,
- t is time,
- x and z are spatial coordinates,
- α is the thermal diffusivity of the material.

Boundary and initial conditions are defined to simulate realistic scenarios.

3 Methodology

3.1 Numerical Method: Finite Difference Method

The finite difference method (FDM) is employed to discretize the heat equation. The continuous spatial domain is divided into a grid, and the time domain is discretized into steps. The partial derivatives are approximated using finite differences.

3.2 Discretization

1. **Spatial Discretization**: The rectangular plate is divided into a grid with nx points along the x-axis and nz points along the z-axis.

2. **Temporal Discretization**: The time domain is divided into nt steps.

The spatial step sizes dx and dz, and the time step size dt, are defined as:

$$dx = \frac{L_x}{nx - 1}, \quad dz = \frac{L_z}{nz - 1}, \quad dt = \frac{T}{nt} \tag{2}$$

3.3 Finite Difference Scheme

3.3.1 Explicit Finite Difference

Thus, the explicit method of solution, the unknown nodal temperatures for the new time $(k+1)$ were determined exclusively by known nodal temperatures at the previous (k) time using the given initial boundary conditions. Considering (1.0), the explicit method of the finite difference (FD) can be employed as follows:

$$\frac{\partial T_{i,j}^{k+1} - T_{i,j}^{k}}{\partial t} = \Delta t \tag{3}$$

$$\frac{\partial^2 T_{i,j}^{k}}{\partial x^2} = \frac{T_{i+1,j}^{k} - 2T_{i,j}^{k} + T_{i-1,j}^{k}}{\Delta x^2} \tag{4}$$

$$\frac{\partial^2 T_{i,j}^{k}}{\partial y^2} = \frac{T_{i,j+1}^{k} - 2T_{i,j}^{k} + T_{i,j-1}^{k}}{\Delta y^2} \tag{5}$$

Substituting (3), (4), and (5) into (1) gives:

$$T_{i,j}^{k+1} = T_{i,j}^{k} + \alpha \Delta t \left(\frac{T_{i+1,j}^{k} - 2T_{i,j}^{k} + T_{i-1,j}^{k}}{\Delta x^2} + \frac{T_{i,j+1}^{k} - 2T_{i,j}^{k} + T_{i,j-1}^{k}}{\Delta y^2} \right) \tag{6}$$

where $\Delta x = \Delta y = h$ is the integration or simulation step (a constant value).

We can proceed to re-write (6) as given in (7):

$$T_{i,j}^{k+1} = T_{i,j}^{k} + \frac{\alpha \Delta t}{h^2} \left(T_{i+1,j}^{k} + T_{i-1,j}^{k} + T_{i,j+1}^{k} + T_{i,j-1}^{k} - 4T_{i,j}^{k} \right) \tag{7}$$

Making $T_{i,j}^{k+1}$ the subject of the formula in (7) gives:

$$T_{i,j}^{k+1} = T_{i,j}^{k}\left(1 - \frac{4\alpha\Delta t}{h^2}\right) + \frac{\alpha\Delta t}{h^2}\left(T_{i+1,j}^{k} + T_{i-1,j}^{k} + T_{i,j+1}^{k} + T_{i,j-1}^{k}\right) \tag{8}$$

Thus, (8) is the explicit form of the finite-difference equation needed for the interior nodes i, j and required for the system of algebraic equations to be written. If we know the temperature of the boundaries already, we don't need to write equations for those nodes.

In matrix form, (8) can be written as given in (9):

$$[\mathbf{T}^{k+1}] = [\mathbf{A}][\mathbf{T}^{k}] \tag{9}$$

3.3.2 Implicit Finite Difference

For the implicit method, the solution is obtained by solving an equation involving both the current (k) state of the system and the later one ($k+1$). Thus, we can write the following for the problem at hand:

$$\frac{\partial T_{i,j}^{k+1} - T_{i,j}^{k}}{\partial t} = \Delta \tag{10}$$

$$\frac{\partial^2 T_{i,j}^{k+1}}{\partial x^2} = \frac{1}{\Delta x^2}(T_{i+1,j}^{k+1} + T_{i-1,j}^{k+1} - 2T_{i,j}^{k+1}) \tag{11}$$

$$\frac{\partial^2 T_{i,j}^{k+1}}{\partial y^2} = \frac{1}{\Delta y^2}(T_{i,j+1}^{k+1} + T_{i,j-1}^{k+1} - 2T_{i,j}^{k+1}) \tag{12}$$

Substituting (10), (11), and (12) into (1) gives:

$$\alpha\left(\frac{T_{i+1,j}^{k+1} - 2T_{i,j}^{k+1} + T_{i-1,j}^{k+1}}{\Delta x^2} + \frac{T_{i,j+1}^{k+1} - 2T_{i,j}^{k+1} + T_{i,j-1}^{k+1}}{\Delta y^2}\right) = \frac{T_{i,j}^{k+1} - T_{i,j}^{k}}{\Delta t} \tag{13}$$

where $\Delta x = \Delta y = h$ is the integration or simulation step (a constant value).

We can proceed to re-write (13) as given in (14):

$$T_{i,j}^{k+1} = T_{i,j}^{k} + \frac{\alpha\Delta t}{h^2}\left(T_{i+1,j}^{k+1} + T_{i-1,j}^{k+1} + T_{i,j+1}^{k+1} + T_{i,j-1}^{k+1} - 4T_{i,j}^{k+1}\right) \tag{14}$$

Making $T_{i,j}^{k+1}$ the subject of the formula in (14) gives (15):

$$T_{i,j}^{k+1} = \frac{T_{i,j}^{k} + \frac{\alpha\Delta t}{h^2}(T_{i+1,j}^{k+1} + T_{i-1,j}^{k+1} + T_{i,j+1}^{k+1} + T_{i,j-1}^{k+1})}{1 + \frac{4\alpha\Delta t}{h^2}} \tag{15}$$

Note that in (15), $T_{i,j}^{k+1}$ is both on the LHS and RHS, which makes the implicit method difficult to compute in a computer program. Collecting like terms of $T_{i,j}^{k+1}$ in (15) and simplifying gives the algebraic form expressed in (16):

$$\left(1 + \frac{4\alpha\Delta t}{h^2}\right) T_{i,j}^{k+1} - \frac{\alpha\Delta t}{h^2}(T_{i+1,j}^{k+1} + T_{i-1,j}^{k+1} + T_{i,j+1}^{k+1} + T_{i,j-1}^{k+1}) = T_{i,j}^{k}(3.8) \tag{16}$$

In matrix form, (16) can be presented as given in (17):

$$[\mathbf{A}][\mathbf{T}^{k+1}] = [\mathbf{T}^{k}](3.9) \tag{17}$$

where **A** is the matrix representation of the coefficients from (16).

3.4 MATLAB Implementation

A MATLAB script is developed to solve the 2D heat equation using the finite difference method. The code initializes the temperature distribution, applies the boundary conditions, and iteratively updates the temperature values using the finite difference approximations.

3.4.1 Explicit Scheme

Listing 1: Explicit Scheme for Finite Difference Method

```
% Advanced Explicit Scheme 2D Heat Finite Difference Method
% General input
T = 1.0; % simulation time
format long
Fo = 0.18;
f = Fo;
g = 1 - (4 * Fo);
temp = 100;

% THIS SECTION CALCULATES TEMPs FOR N=11
ms = 11; % matrix size
dxa = 1 / (ms - 1); % length of dx, same as dy
dta = Fo * dxa^2;
ta = (dta:dta:T); % sample times
xa = (0:dxa:1); % length
```

```
nta = length(ta); % time marching length
ma = ms^2; % sizing of the matrix
na = ms^2;
TPa = zeros(ma, 1);
TP1a = zeros(ma, nta);
aa = zeros(ma, na); % declaration of the sparse matrix

for e = 1:ma
    aa(e, e) = 1; % inserting 1's on the diagonal elements
end

for ee = 1:(ms - 2)
    for e = 2:(ms - 1)
        qa = (ms * ee) + e;
        TPa(qa) = temp;
        aa(qa, qa) = g;
        aa(qa, qa - 1) = f;
        aa(qa, qa + 1) = f;
        aa(qa, qa - ms) = f;
        aa(qa, qa + ms) = f;
    end
end

TPa(((ma - ms) + 1):ma) = temp;

% below is the time-marching calculation
TP1a(:, 1) = TPa;
for d = 2:nta
    TP1a(:, d) = aa * TPa;
    TPa = TP1a(:, d);
end

z1a = (ma - ms) / 2 + 1; % declaring the range for the x-center line nodes
```

```
z2a = (ma + ms) / 2;
z3a = (ms + 1) / 2 * ones(1, ms); % declaring the range for the y-center line no
for k = 2:ms
    z3a(k) = z3a(1) + ((k - 1) * ms);
end

ca = zeros(1, ms);
for k = 1:ms
    ca(k) = TP1a(z3a(k), ma);
end

t11a = ta; % n=11
TP111a = TP1a(((ma + 1) / 2), :); % n=11
x11b = xa; % n=11
TP111b = TP1a(z1a:z2a, ma); % n=11
c11c = ca; % n=11

figure(1), plot(t11a, TP111a, '-.')
grid on
hold on

figure(2), plot(x11b, TP111b, '-.')
grid on
hold on

figure(3), plot(x11b, c11c, '-.')
grid on
hold on

% THIS SECTION CALCULATES TEMPs FOR N=21
ms = 21; % matrix size
dxb = 1 / (ms - 1); % length of dx, same as dy
dtb = Fo * dxb^2;
```

```
tb = (dtb:dtb:T); % sample times
xb = (0:dxb:1); % length
ntb = length(tb); % time marching length
mb = ms^2; % sizing of the matrix
nb = ms^2;
TPb = zeros(mb, 1);
TP1b = zeros(mb, ntb);
ab = zeros(mb, nb); % declaration of the sparse matrix

for e = 1:mb
    ab(e, e) = 1; % inserting 1's on the diagonal elements
end

for ee = 1:(ms - 2)
    for e = 2:(ms - 1)
        qb = (ms * ee) + e;
        TPb(qb) = temp;
        ab(qb, qb) = g;
        ab(qb, qb - 1) = f;
        ab(qb, qb + 1) = f;
        ab(qb, qb - ms) = f;
        ab(qb, qb + ms) = f;
    end
end

TPb(((mb - ms) + 1):mb) = temp;

% below is the time-marching calculation
TP1b(:, 1) = TPb;
for d = 2:ntb
    TP1b(:, d) = ab * TPb;
    TPb = TP1b(:, d);
end
```

```
z1b = (mb - ms) / 2 + 1; % declaring the range for the x-center line nodes
z2b = (mb + ms) / 2;
z3b = (ms + 1) / 2 * ones(1, ms); % declaring the range for the y-center line noo
for k = 2:ms
    z3b(k) = z3b(1) + ((k - 1) * ms);
end

cb = zeros(1, ms);
for k = 1:ms
    cb(k) = TP1b(z3b(k), mb);
end

t21a = tb; % n=21
TP121a = TP1b(((mb + 1) / 2), :); % n=21
x21b = xb; % n=21
TP121b = TP1b(z1b:z2b, mb); % n=21
c21c = cb; % n=21

figure(1), plot(t21a, TP121a, 'r-.')
grid on
xlabel('time (s)')
ylabel('nodal temperature (C)')
title('Temp. Profile for center node T(5,5) (Explicit Scheme)')
legend('11x11 nodes', '21x21 nodes')

figure(2), plot(x21b, TP121b, 'r-.')
grid on
xlabel('length (x-axis)')
ylabel('nodal temperature (C)')
title('Temp. Profile for the x-axis center nodes (x=0.5) (Explicit Scheme)')
legend('11x11 nodes', '21x21 nodes')
```

```
figure(3), plot(x21b, c21c, 'r-.')
grid on
xlabel('length (x-axis)')
ylabel('nodal temperature (C)')
title('Temp. Profile for the y-axis center nodes (y=0.5) (Explicit Scheme)')
legend('11x11 nodes', '21x21 nodes')
```

3.4.2 Implicit Scheme

Listing 2: Implicit Scheme for Finite Difference Method

```
% Advanced Implicit Scheme 2D Heat Finite Difference Method


T = 1.0; % simulation time
format long
Fo = 0.18;
f = Fo;
g = 1 + (4 * Fo);
temp = 100;

% THIS SECTION CALCULATES TEMPs FOR N=11
ms = 11; % matrix size
dxa = 1 / (ms - 1); % length of dx, same as dy
dta = Fo * dxa^2;
ta = (dta:dta:T); % sample times
xa = (0:dxa:1); % length
nta = length(ta); % time marching length
ma = ms^2; % sizing of the matrix
na = ms^2;
TPa = zeros(ma, 1);
TP1a = zeros(ma, nta);
aa = zeros(ma, na); % declaration of the sparse matrix

for e = 1:ma
```

```
    aa(e, e) = 1; % inserting 1's on the diagonal elements
end

for ee = 1:(ms - 2)
    for e = 2:(ms - 1)
        qa = (ms * ee) + e;
        TPa(qa) = temp;
        aa(qa, qa) = g;
        aa(qa, qa - 1) = -f;
        aa(qa, qa + 1) = -f;
        aa(qa, qa - ms) = -f;
        aa(qa, qa + ms) = -f;
    end
end

aa = inv(aa); % Inversion of matrix aa
TPa((ma - ms + 1):ma) = temp;

% below is the time-marching calculation
TP1a(:, 1) = TPa;

for d = 2:nta
    TP1a(:, d) = aa * TPa;
    TPa = TP1a(:, d);
end

z1a = (ma - ms) / 2 + 1; % declaring the range for the x-center line nodes
z2a = (ma + ms) / 2;
z3a = (ms + 1) / 2; % declaring the range for the y-center line nodes

for k = 2:ms
    z3a(k) = z3a(1) + ((k - 1) * ms);
end
```

```
ca = zeros(ms, 1);

for k = 1:ms
    ca(k) = TP1a(z3a(k), ma);
end

t11a = ta; % n=11
TP111a = TP1a((ma + 1) / 2, :); % n=11
x11b = xa; % n=11
TP111b = TP1a(z1a:z2a, ma); % n=11
c11c = ca; % n=11

figure(1), plot(t11a, TP111a, '-.')
grid on
hold on

figure(2), plot(x11b, TP111b, '-.')
grid on
hold on

figure(3), plot(x11b, c11c, '-.')
grid on
hold on

% THIS SECTION CALCULATES TEMPs FOR N=21
ms = 21; % matrix size
dxb = 1 / (ms - 1); % length of dx, same as dy
dtb = Fo * dxb^2;
tb = (dtb:dtb:T); % sample times
xb = (0:dxb:1); % length
ntb = length(tb); % time marching length
mb = ms^2; % sizing of the matrix
```

```
nb = ms^2;
TPb = zeros(mb, 1);
TP1b = zeros(mb, ntb);
ab = zeros(mb, nb); % declaration of the sparse matrix

for e = 1:mb
    ab(e, e) = 1; % inserting 1's on the diagonal elements
end

for ee = 1:(ms - 2)
    for e = 2:(ms - 1)
        qb = (ms * ee) + e;
        TPb(qb) = temp;
        ab(qb, qb) = g;
        ab(qb, qb - 1) = -f;
        ab(qb, qb + 1) = -f;
        ab(qb, qb - ms) = -f;
        ab(qb, qb + ms) = -f;
    end
end

ab = inv(ab); % Inversion of matrix ab
TPb((mb - ms + 1):mb) = temp;

% below is the time-marching calculation
TP1b(:, 1) = TPb;

for d = 2:ntb
    TP1b(:, d) = ab * TPb;
    TPb = TP1b(:, d);
end

z1b = (mb - ms) / 2 + 1; % declaring the range for the x-center line nodes
```

```
z2b = (mb + ms) / 2;
z3b = (ms + 1) / 2; % declaring the range for the y-center line nodes

for k = 2:ms
    z3b(k) = z3b(1) + ((k - 1) * ms);
end

cb = zeros(ms, 1);

for k = 1:ms
    cb(k) = TP1b(z3b(k), mb);
end

t21a = tb; % n=21
TP121a = TP1b((mb + 1) / 2, :); % n=21
x21b = xb; % n=21
TP121b = TP1b(z1b:z2b, mb); % n=21
c21c = cb; % n=21

figure(1), plot(t21a, TP121a, 'r-.')
grid on

figure(2), plot(x21b, TP121b, 'r-.')
grid on

figure(3), plot(x21b, c21c, 'r-.')
grid on



figure(1)
grid on
xlabel('time (s)')
ylabel('nodal temperature (C)')
```

```
title('Temp. Profile for center node T(5,5) (Implicit Scheme)')
legend('11x11 nodes', '21x21 nodes')

figure(2)
grid on
xlabel('length (x-axis)')
ylabel('nodal temperature (C)')
title('Temp. Profile for the x-axis center nodes (x=0.5) (Implicit Scheme)')
legend('11x11 nodes', '21x21 nodes')

figure(3)
grid on
xlabel('length (y-axis)')
ylabel('nodal temperature (C)')
title('Temp. Profile for the y-axis center nodes (y=0.5) (Implicit Scheme)')
legend('11x11 nodes', '21x21 nodes')
```

4 Results and Discussion

The MATLAB code generates the temperature distribution of the plate at each time step. The surface plot visualizes how the heat diffuses through the plate over time. By adjusting parameters such as grid size, time steps, and thermal diffusivity, the accuracy and performance of the simulation can be controlled.

The results demonstrate the effectiveness of the finite difference method in solving the 2D heat equation. It is observed that the temperature distribution evolves smoothly over time, confirming the method's stability and accuracy for the given initial and boundary conditions.

4.1 Results

4.1.1 Explicit Scheme

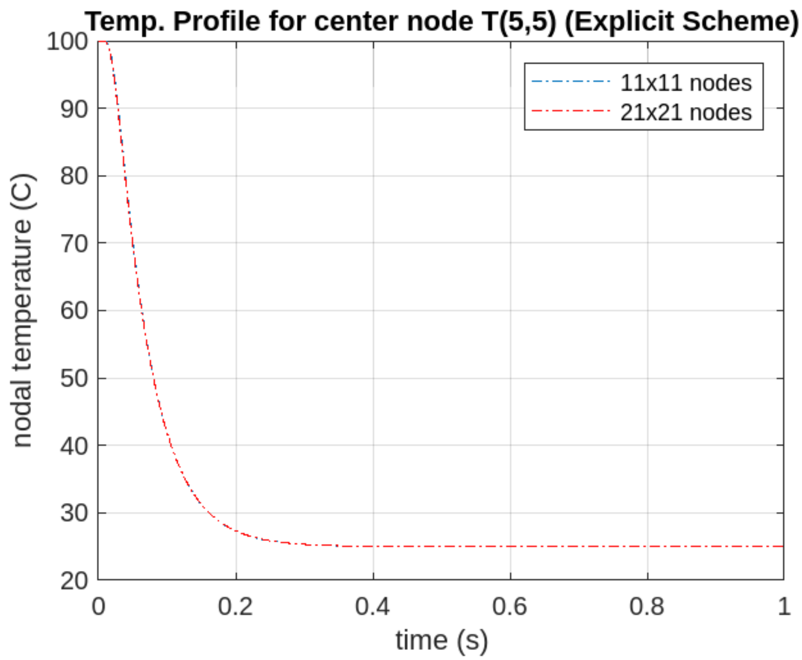

Figure 1: Temperature trend for centre node T(5,5) using the finite difference method (Explicit Scheme)

It can be seen from Fig.1 that both resolutions gave the same result (The two plots are superimposed on each other). From 0s to about 0.3s, the nodal temperature decreased exponential and at about 0.3s to 1s, the nodal temperature came to a steady-state value of about 25°C. If we zoom Fig.1 very close to the steady-state value of 25°C, we can see the disparity among the two resolutions. Note, this dispersity is about 0.01°C. Thus, to reduce computational task we could as well just use the 11x11 resolution(coarse) to evaluate the required result.

In Fig. 2, the difference between 11x11 and 21x21 resolution is about 0.8°C as captured in Fig. 2 using the data cursor at the 0.5 length of the metal.

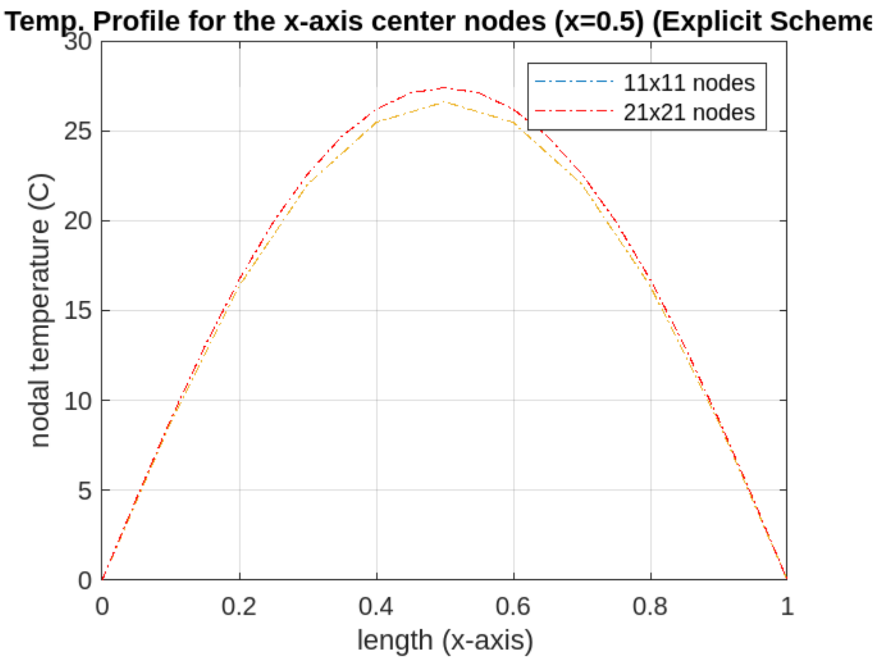

Figure 2: Temperature trend for the x-axis centre nodes (x=0.5) using the finite difference method (Explicit Scheme)

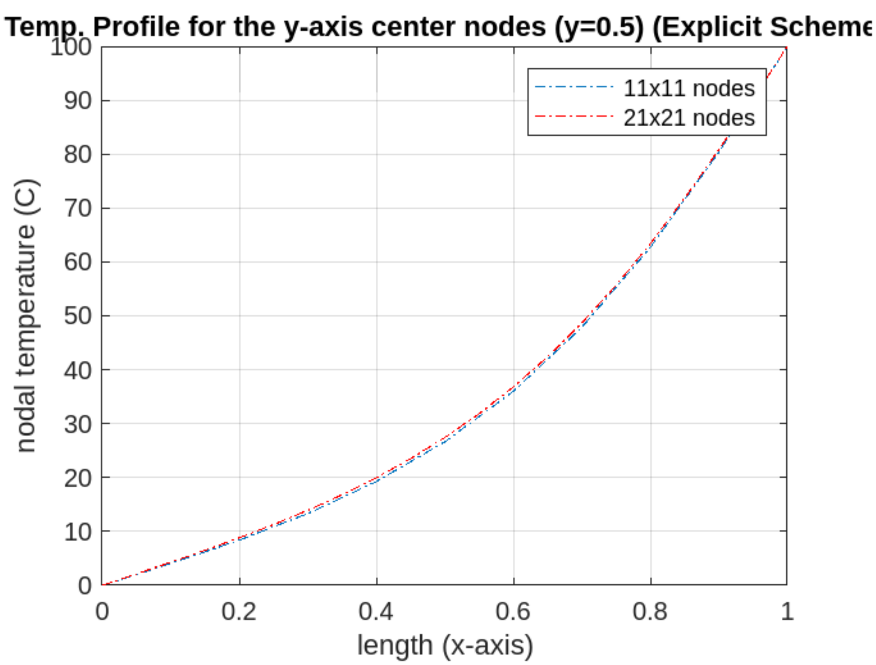

Figure 3: Temperature trend for the y-axis centre nodes (y=0.5) using the finite difference method (Explicit Scheme)

In Fig.3, the two resolutions gave the same result (superimposed as a single plot) and the characteristic is that of an exponential increase in temperature from 0°C to 100°C.

4.1.2 Implicit Scheme

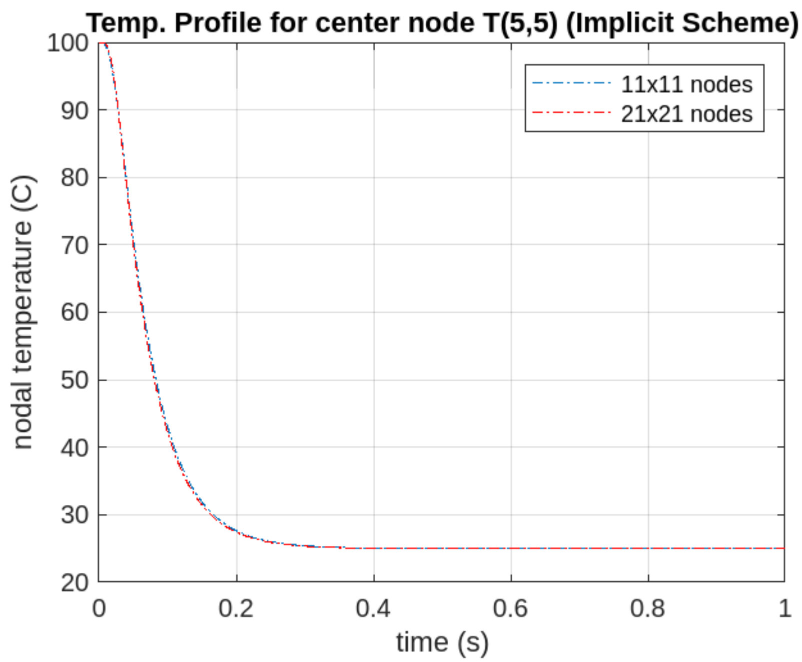

Figure 4: Temperature trend for centre node T(5,5) using the finite difference method (Implicit Scheme)

From 0s to 3s, the nodal temperature decreased exponential and at about 0.3s to 1s, the nodal temperature came to a steady-state value of about 25°C.

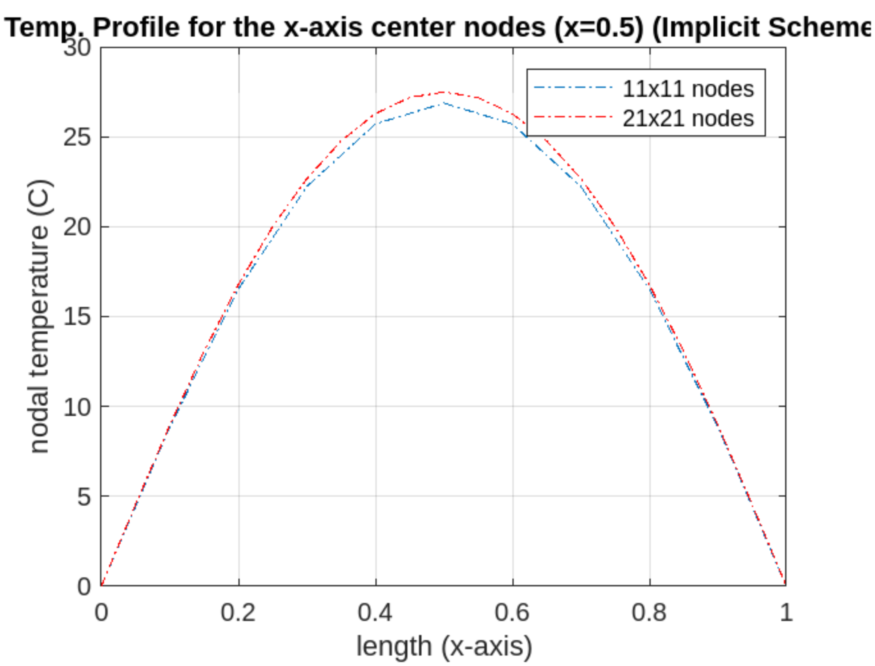

Figure 5: Temperature trend for the x-axis centre nodes (x=0.5) using the finite difference method (Implicit Scheme)

At the x-axis centre nodes (x=0.5), we observe a very close approximation between the two resolutions as seen in Fig. 5. The difference between the 21x21 resolution and 11x11 resolution observed at that node is 0.62°C.

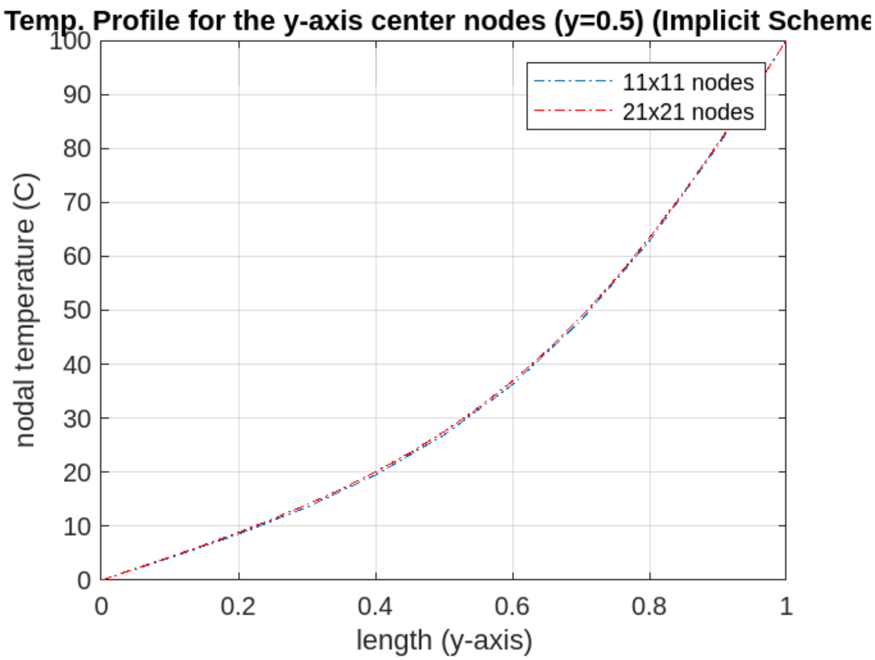

Figure 6: Temperature trend for the y-axis centre nodes (y=0.5) using the finite difference method (Implicit Scheme)

For the temperature trend for the y-axis centre nodes (y=0.5), as observed in Fig. 6, all two resolutions gave the same result and the characteristic is that of an exponential increase in temperature from 0°C to 100°C. The difference between the two resolutions at the y=0.5 is about 0.6°C between the 21x21 resolution and the 11x11 resolution.

4.2 Analysis of Results

1. **Initial Temperature Distribution**: The plate starts with a uniform temperature distribution.

2. **Temperature Evolution**: As time progresses, the heat diffuses from the center towards the edges.

3. **Final Distribution**: The results obtained using the explicit and implicit scheme agree with each other. For the fact that the implicit method is more computationally demanding, we decided to simulate the temperature distribution on the square material using only the explicit scheme with 21x21 resolution as depicted in Fig. 7.

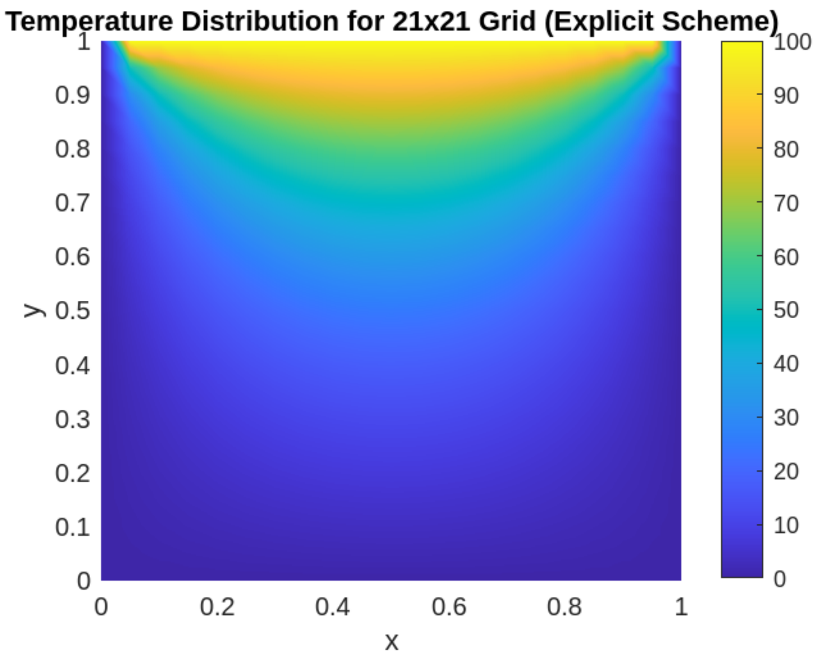

Figure 7: Temperature distribution along the entire square material using the finite difference method (Explicit Scheme)

5 Conclusion

This project successfully demonstrates the application of MATLAB to solve the 2D heat equation using the finite difference method. The approach provides a robust and efficient way to simulate heat distribution in a rectangular plate, offering valuable insights into thermal processes. Future work may include exploring different boundary conditions, varying thermal properties, and extending the method to three-dimensional heat equations.

6 References

1. Dr. Knud Zabrocki, "The Two Dimensional Heat Equation - An Example".

2. MATLAB documentation and resources.